Analiza cerintelor

Se cere aplicarea unui filtru peste o imagine reprezentata cu ajutorul unei matrici. Acest lucru trebuie realizat fara a avea o noua matrice pentru rezultat. Algoritmul trebuie implementat in Java si C++

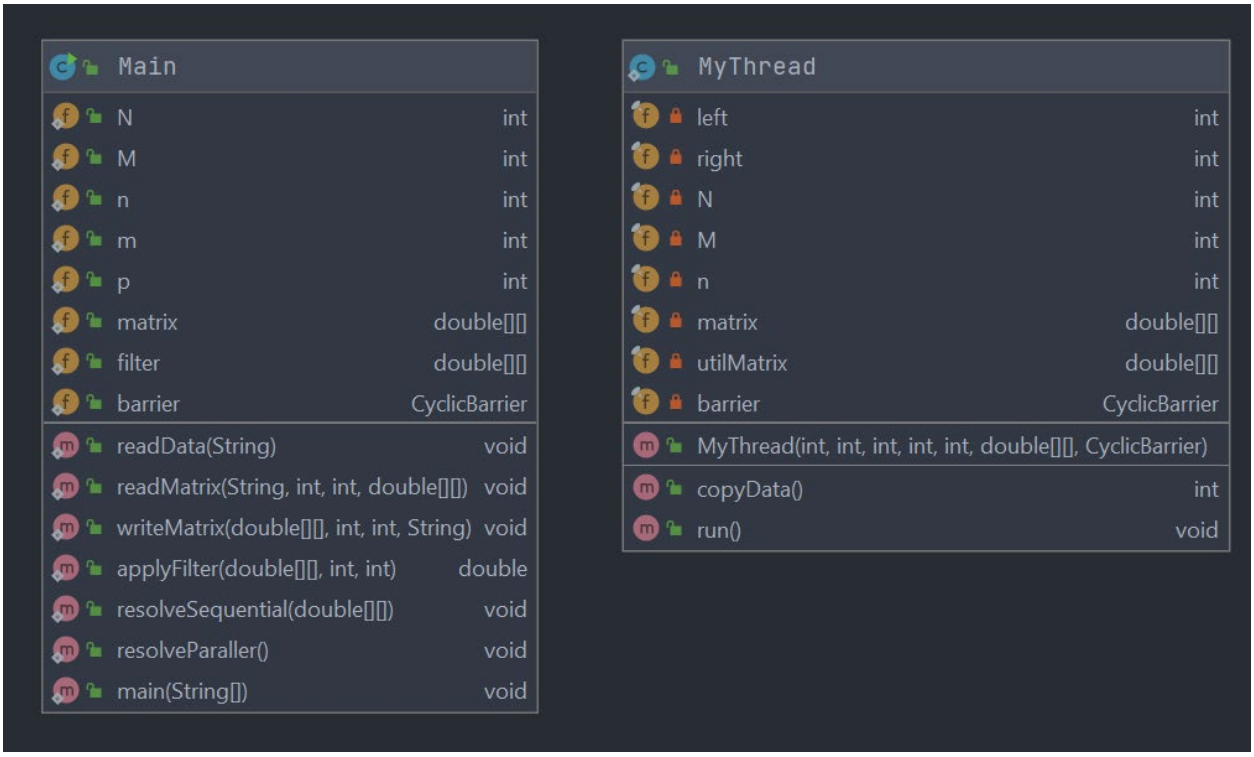
Proiectare

Sunt citite din fisier: datele, imagine si filtrul.

Algoritmul foloseste distributia pe linii. Se impart datele egal la numarul de threaduri. Unele threaduri pot avea mai multe date in cazul in care apare rest la impartire.

Fiecare thread salveaza intr-o matrice auxiliara datele de care are nevoie, dar si date de la threadurile vecine. Este utilizata o bariera pentru a lasa toate threadurile sa salveze datele inainte de modificare. In urmatorul pas fiecare thread aplica filtrul pe datele deja salvate si pastreaza rezultatul in matricea originala.

In final matricea este scrisa in fisier.



Java

Tip Matrice	Nr threads	Timp executie
N = M = 10 n = m = 3	1	1.3
	2	3.7
N = M = 1000 n = m = 5	1	151.4
	2	107.4
	4	117.3
	8	154
	16	195.8
N = 10 M = 10000 n = m = 5	1	73.4
	2	52.4
	4	58.9
	8	55.5
	16	80.6
N = 10000 M = 10 n = m = 5	1	41.7
	2	42.5
	4	39.7
	8	41.8
	16	41

C++

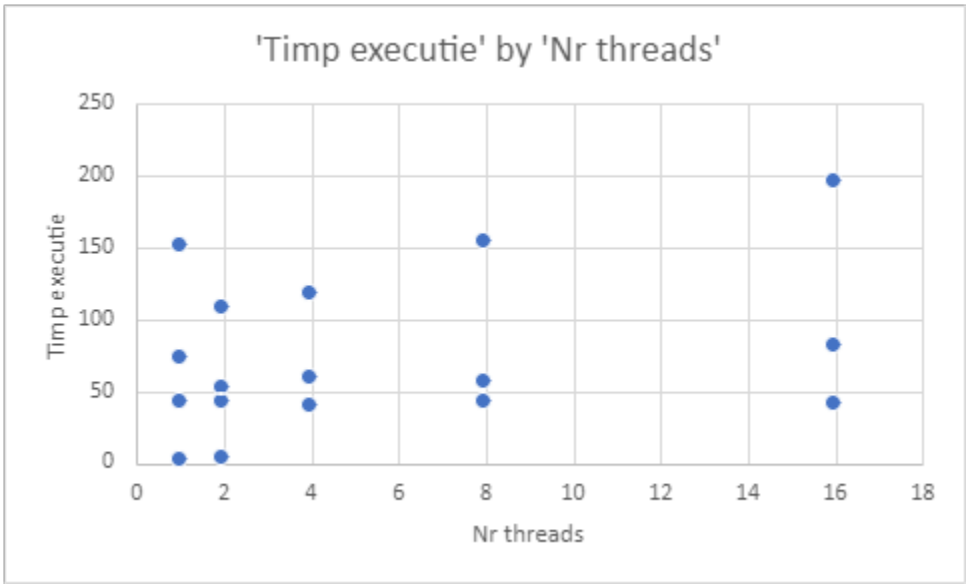
Tip Matrice	Nr threads	Timp executie
N = M = 10 n = m = 3	1	0.105
	2	0.499
N = M = 1000 n = m = 5	1	49.303
	2	29.392
	4	25.878
	8	25.987
	16	26.177
N = 10 M = 10000 n = m = 5	1	5.724
	2	3.496
	4	4.083
	8	4.088
	16	4.997
N = 10000 M = 10 n = m = 5	1	6.150
	2	4.602
	4	3.579
	8	4.684
	16	4.383

Analiza

- Timpii de executie realizati in C++ sunt mai buni in toate cazurile decat cei realizati in Java

- Pentru un volum mai mare de date, timpii de executie au scazut direct proportional cu cresterea numarului de threaduri
- Cei mai buni timpi de executie sunt obtinuti mereu atunci cand sunt folosite 4 threaduri

Java



C++